FUZZY NEURAL NETS

There is no doubt that neurocomputing and fuzzy set technology were dominant information technologies of the 1990s. The dominant paradigm of neurocomputing (1) is concerned with parallel and distributed processing realized by a vast number of simple processing units known as (artificial) neurons. The neural networks are universal approximators, meaning that they can approximate continuous relationships to any desired accuracy. This feature is intensively exploited in a vast number of applications of neural networks, in areas such as pattern recognition, control, and system identification. The underlying philosophy of fuzzy sets is that of a generalization of set theory with an intent of formalization of concepts with gradual boundaries (2). Through the introduction of fuzzy sets one develops a suitable conceptual and algorithmic framework necessary to cope with the most suitable level of information granularity. All constructs arising from the ideas of fuzzy sets hinge on the notion of information granularity and linguistic nonnumeric information, in particular. It becomes apparent that the technologies of fuzzy sets and neural networks are complementary: fuzzy sets deliver a suitable conceptual framework while neural networks furnish us with all necessary learning capabilities.

The fuzzy set–neural networks synergy has already led to a number of interesting architectures that are usually referred to as fuzzy–neural systems or fuzzy neural networks. These are the models, the development of which heavily depends upon fuzzy sets and neurocomputing. The contribution

of the two technologies could vary from case to case. In some cases, we envision a significant dominance of fuzzy sets with some additional learning slant supported by neural networks. In other scenarios, one can witness structures that are essentially neural networks with some structural enhancements coming from the theory of fuzzy sets. In a nutshell, the existing diversity of such approaches calls for their systematic treatment that definitely helps us understand the benefits of the symbiosis and make the design of such systems more systematic. This study is organized in a way that unveils the main architectural and learning issues of synergy between neural networks and fuzzy sets. The agenda of this article is twofold:

- First, we propose a general taxonomy of hybrid fuzzy–neural topologies by studying various temporal and architectural aspects of this symbiosis.
- Second, our intent is to review some representative examples of hybrid structures that illustrate the already introduced typology thoroughly.

NEUROCOMPUTING IN FUZZY SET TECHNOLOGY

Generally speaking, in the overall fuzzy set–neural network hybrid methodology, neural networks are providers of useful computational and learning facilities. Fuzzy sets, as based on the mechanisms of set theory and multivalued logic, are chiefly preoccupied with the variety of aspects of knowledge representations. At the same time they tend to be somewhat weaker as far as their processing capabilities are concerned (interestingly enough, this claim becomes valid in the case of all constructs originating from set theory). In particular, set-theoretic operations do not cope explicitly with repetitive information and cannot reflect this throughout their outcomes—the most evident examples arises in terms of highly noninteractive maximum and minimum operations. Being more specific, the result of the minimum (or maximum) operation relies on the extreme argument and does not tackle the remaining elements. Say min (0.30, 0.9, 0.95, 0.87, 0.96) is the same as the one of the expression min (0.30, 0.41, 0.32, 0.33, 0.31).

There are a number of instances in which neural networks are used directly to support or realize computing with fuzzy sets. In general, in most of these cases, neural networks are aimed at straightforward computing through the utilization of membership values. Similarly, there are various approaches spearheaded along the line of the development of neural networks with the intent of processing fuzzy information. In this case there is not too much direct interaction and influence originating from the theory of fuzzy sets. It is essentially a way in which neural networks are aimed at the calibration of fuzzy sets—the construction of their membership functions is completed in the setting of numeric data available at hand.

Fuzzy Sets in the Technology of Neurocomputing

The key role of fuzzy sets is to enhance neural networks by incorporating knowledge-oriented mechanisms. Generally speaking, these knowledge-based enhancements of neural networks are threefold:

- Preprocessing of training data that could easily lead to the improvement in learning and/or enhanced robustness characteristics of the network
- Enhancements of specific training procedures through knowledge-based learning schemes (including learning metarules)
- Linguistic interpretation of results produced by neural networks

Each of these areas have specific and highly representative instances. We review them to expose the reader to the very nature of some important functional links between fuzzy sets and neural networks.

Fuzzy Sets in the Preprocessing and Utilization of Training Data

The function of fuzzy sets in this particular framework is to deliver an interface between the data (environment) and the neural network regarded primarily as a processing vehicle. As visualized in Fig. 1, the original data are transformed within the framework of the fuzzy set interface: the resulting format could be very different from the one encountered in the original environment.

The intent of the interface is to expose the network to the most essential features of the data that need to be captured through the subsequent mechanisms of learning. These features are usually revealed as a part of the underlying domain knowledge. The notion of a cognitive perspective develops a suitable learning environment. By selecting a collection of so-called linguistic landmarks (2,3), one can readily meet several important objectives:

- Performing a nonlinear normalization of the training data. By transforming any real data, any pattern $x \in \mathbb{R}^n$ becomes converted into the corresponding element of a highly dimensional unit hypercube.
- Defining a variable (as opposed to fixed) processing resolution carried out within the resulting neural networks.
- Coping with uncertainty in the training data.

Let us briefly elaborate on the nature of these enhancements.

Nonlinear Data Normalization. For each coordinate (variable) we define c linguistic terms. These are denoted en block

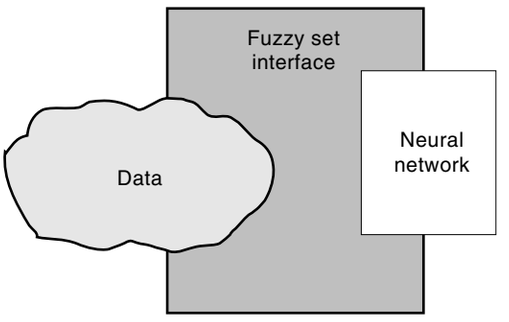

Figure 1. Fuzzy sets in interfacing neural networks with data environment: transforming data into a format assuring computational efficiency of neurocomputation.

as $\mathscr{A} = \{A_1, A_2, \ldots, A_c\}$ for the first coordinate, $\mathscr{B} = \{B_1, B_2, \ldots, B_c\}$ for the second, etc. Then the linguistic preprocessing $\mathscr{P}$ carries out the mapping of the form

$$\mathscr{P} : \mathbb{R}^n \rightarrow [0, 1]^{nc} \tag{1}$$

More specifically, a numeric input x invokes (activates) a series of linguistic terms $A_1, A_2, \ldots, A_c, B_1, B_2, \ldots$, etc. As we are concerned with "n" dimensional inputs with "c" labels (fuzzy sets) associated each of them, we end up with $n * c$ activation levels situated in the unit interval. Or, in other words, the results of this nonlinear transformation are located in the $n * c$-dimensional unit hypercube. Observe also that this preprocessing serves as a useful nonlinear data normalization. In contrast, the commonly exploited linear transformation defined as

$$\frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2}$$

(where $x_{\min}$ and $x_{\max}$ are the bounds of the variable) does not exhibit any nonlinear effect.

The positive effect of data normalization has often been underlined in many studies on neural networks. The normalization is always recommended, especially if the ranges of the individual variables are very distinct, say [0,0.05] *vis-à-vis* $[10^6, 10^8]$. The direct use of rough (unscaled) data could easily lead to a completely unsuccessful learning. The nonlinear effect conveyed by Eq. (1) stems from the nonlinear membership functions of the linguistic terms.

The linguistic preprocessing increases the dimensionality of the problem; however, it could also decrease the learning effort. The similar speedup effect in training is commonly observed in radial basic function (RBF) neural networks (4,5). The improvement in the performance achieved in this setting stems from the fact that the individual receptive fields modeled by the RBFs identify homogeneous regions in the multidimensional space of input variables. Subsequently, the updates of the connections of the hidden layers are less demanding, as a preliminary structure has been already established and the learning is oriented towards less radical changes of the connections and practically embarks on some calibration of the receptive fields. By modifying the form of the RBFs themselves, some regions exhibiting a significant variability of the approximated function are made smaller so that a single linear unit can easily adjust. Similarly, the regions over which the approximated function does not change drastically can be made quite large by adapting radial basis functions of lower resolution. In general, this concept leads to the concept of multiresolutionlike (fractal-oriented) neural networks.

Variable Processing Resolution. By defining the linguistic terms (modeling landmarks) and specifying their distribution along the universe of discourse we can orient (focus) the main learning effort of the network. To clarify this idea, let us refer to Fig. 2.

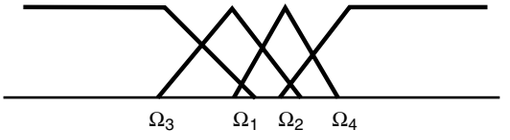

Figure 2. Fuzzy quantization (partition, discretization) delivered by linguistic terms. Note a diversity of information granularity (variable processing resolution) captured by the respective linguistic terms.

The partition of the variable through $\mathscr{A}$ assigns a high level of information granularity to some regions (say Ω_1 and Ω_2) and sensitizes the learning mechanism accordingly. On the other hand, the data points falling under Ω_3 are regarded internally (at the level they are perceived by the networks) as equivalent (by leading to the same numeric representation in the unit hypercube).

Uncertainty Representation. The factor of uncertainty or imprecision can be quantified by exploiting some uncertainty measures as introduced in the theory of fuzzy sets. The underlying rationale is to equip the internal format of information available to the network with some indicators describing how uncertain a given piece of data is. Considering possibility and necessity measures this quantification is straightforward: once $\text{Poss}(X, A_k) \neq \text{Nec}(X, A_k)$, then X is regarded uncertain (the notion of uncertainty is also context-sensitive and depends on A_k) (6). For numerical data one always arrives at the equality of these two measures that underlines the complete certainty of X. In general, the higher the gap between the possibility and necessity measures, $\text{Poss}(X, A_k) = \text{Nec}(X, A_k) + \delta$, the higher the uncertainty level associated with X. The uncertainty gap attains its maximum for $\delta = 1$.

The way of treating the linguistic term makes a real difference between the architecture outlined above and the standard RBF neural networks. The latter ones do not have any provisions to deal with and quantify uncertainty. The forms of the membership function (RBFs) are very much a secondary issue. In general, one can expect that the fuzzy sets used therein can exhibit a variety of forms (triangular, Gaussian, etc.), while RBFs are usually more homogeneous (e.g., all assume Gaussian-like functions). Furthermore, there are no specific restrictions on the number of RBFs used as well as their distribution across the universe of discourse. For fuzzy sets one restricts this number to a maximum of 9 terms (more exactly, 7 ± 2); additionally we make sure that the fuzzy sets are kept distinct and thus retain a clear semantic identity that supports their interpretation.

Knowledge-Based Learning Schemes

Fuzzy sets influence neural networks as far as learning mechanisms and interpretation of the constructed networks are constructed. The set of examples discussed in this section illustrates this point.

Metalearning and Fuzzy Sets

Even though guided by detailed gradient-based formulas, the learning of neural networks can be enhanced by making use of some domain knowledge acquired via intense experimentation (learning). By running a mixture of successful and unsuccessful learning sessions one can gain a qualitative knowledge on what an efficient leaning scenario should look like.

In particular, some essential qualitative associations can be established by linking the performance of the learning process and the parameters of the training scheme being utilized. Two detailed examples follow.

The highly acclaimed backpropagation (BP) scheme used in training multilevel neural networks is based upon the gradient of the performance index (objective index) Q. The basic update formula reads now as

$$w_{ij} = w_{ij} - \alpha \frac{\partial Q}{\partial w_{ij}}$$

where w_{ij} stands for a connection (weight) between the two neurons (i and j). The positive learning rate is denoted by α. Similarly $\partial Q/\partial w_{ij}$ describes a gradient of Q expressed with respect to w_{ij}. Obviously, higher values of α result in more profound changes (updates) of the connection. Higher values of α could result in faster learning that, unfortunately, comes at the expense of its stability (oscillations and overshoots in the values of Q). Under such circumstances, one may easily end up with the diverging process of learning. After a few learning sessions one can easily reveal some qualitative relationships that could be conveniently encapsulated in the form of "if–then" rules (7).

if there are changes of $Q(\Delta Q)$, then there are changes in $\Delta\alpha$

A collection of such learning rules (metarules) is shown in Table 1.

These learning rules are fairly monotonic (yet not symmetric) and fully comply with our intuitive observations when it comes to the representation of the supervisory aspects of the learning procedures in neural networks. In general, any increase in Q calls for some decrease of α; when Q decreases, then the increases in α need to be made more conservative. The linguistic terms in the corresponding rules are defined in the space (universe) of changes of Q (antecedents) and a certain subset of [0,1] (conclusions). Similarly, the BP learning scheme can be augmented by taking into account a momentum term; the primary intent of this expansion is to suppress eventual oscillations of the performance index or reduce its amplitude. This makes the learning more stable, yet adds one extra adjustable learning parameter in the update rule itself.

The learning metarules rules can be formulated at the level of some critical parameters of the networks. The essence of the ensuing approach is to modify activation functions of the neurons in the network. Consider the sigmoid nonlinearity (that is commonly encountered in many neural architectures)

$$y = \frac{1}{1 + \exp(-\gamma u)}$$

We assume that the steepness factor of the sigmoid function (γ) is modifiable. As the changes of the connections are evidently affected by this parameter (γ), we can easily set up metarules of the form:

if the performance index is Q_o then γ is γ_o

Table 1. BP-Oriented Learning Rules*

ΔQ	$\Delta\alpha$
NB	PB
NM	PM
NS	PS
Z	Z
PS	NM
PM	NB
PB	NB

*NB, negative big; NM, negative medium; NS, negative small; Z, zero; PS, positive small; PM, positive medium; PB, positive big.

As before, it is intuitively straightforward to set up a collection of the detailed learning rules. Summing up, we highlight two crucial design issues:

- The considered approach is fully experimental. The role of fuzzy sets is to represent and to summarize the available domain knowledge properly.
- While the control protocol (rules) seems to be universal, the universes of discourse should be modified according to the current application (problem). In other words, the basic linguistic terms occurring therein need to be adjusted (calibrated). The realization of this phase calls for some additional computational effort, which could somewhat offset the benefits originating from the availability of the domain knowledge.

Fuzzy Clustering in Revealing Relationships within Data

In the second approach the domain knowledge about learning is acquired through some preprocessing of training data prior to running any specific learning scheme. This is the case in the construction known as a *fuzzy perceptron* (8). In an original setting, a single-layer perceptron is composed of a series of linear processing units equipped with the threshold elements. The basic perception-based scheme of learning is straightforward. Let us start with a two-category multidimensional classification problem. If the considered patterns are linearly separable, then there exists a linear discriminant function f such that

$$f(\mathbf{x},\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} \begin{cases} > 0 \text{ if } \mathbf{x} \text{ is in class 1} \\ < 0 \text{ if } \mathbf{x} \text{ is in class 2} \end{cases}$$

where $\mathbf{x},\mathbf{w} \in \mathbb{R}^{n+1}$. The dimensionality of the original space of patterns ($\mathbb{R}^n$) has been increased due to a constant term standing in the linear discriminant function, say $f(\mathbf{x},\mathbf{w}) = \mathbf{w}_o \cdot 1 + \mathbf{w}_1\mathbf{x}_1 + \mathbf{w}_2\mathbf{x}_2 + \cdot\ \cdot\ \cdot + \mathbf{w}_n\mathbf{x}_n$.

After multiplying the class 2 patterns by -1, we obtain the system of positive inequalities

$$f(\mathbf{x}_k,\mathbf{w}) > 0$$

where $k = 1,2,\ldots,N$. The learning concerns a determination of the connections ($\mathbf{w}$) so that all inequalities are made positive. The expression $f(\mathbf{x}_k,\mathbf{w}) = 0$ defines a hyperplane partitioning the patterns; all class 1 patterns are located at the same side of this hyperplane. Assume that x_k's are *linearly* separable. The perception algorithm (shown as follows) guarantees that the discriminating hyperplane (vector $\mathbf{w}$) is found in a *finite* number of steps.

do for all vectors $\mathbf{x}_k$, $= 1,2,\ldots,N$
  if $\mathbf{w}^{\mathrm{T}}\mathbf{x}_k \le 0$, then update the weights (connections)
  $\mathbf{w} = \mathbf{w} + c\mathbf{x}_k$
  $c > 0$
end;
the loop is repeated until no updates of $\mathbf{w}$ occur

The crux of the preprocessing phase as introduced by Keller and Hunt (8) is to carry out the clustering of data and determine the prototypes of the clusters as well as compute the class membership of the individual patterns. The ensuing membership values are used to monitor the changes. Let u_{ik}

and u_{2k} be the membership grades of the kth pattern. Definitely, $u_{1k} + u_{2k} = 1$. The outline of the learning algorithm is the same as before. The only difference is that the updates of the weights are governed by the expression

$$\mathbf{w} = \mathbf{w} + c\mathbf{x}_k|u_{1k} - u_{2k}|^p$$

where $p > 1$. These modifications depend very much on the belongingness of the current pattern in the class. If $u_{1k} = u_{2k} = 0.5$, then the correction term is equal to zero and no update occurs. On the other hand, if $u_{1k} = 1$, then the updates of the weights are the same as those encountered in the original perceptron algorithm.

In comparison to the previous approaches, the methods stemming from this category require some extra computing (preprocessing) but relieve us from the calibration of the linguistic terms (fuzzy sets) standing in the learning metarules.

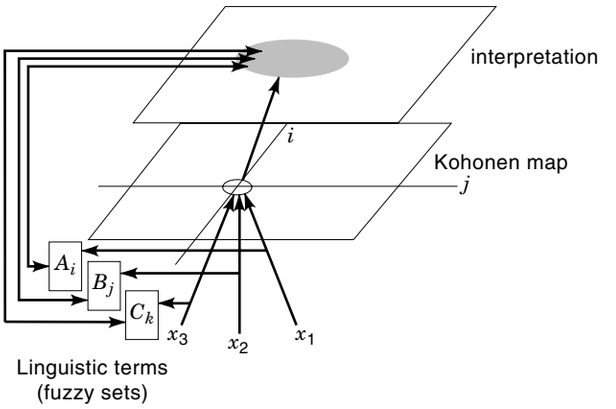

Figure 3. Self-organizing (Kohonen) map and its linguistic interpretation produced by an additional interpretation layer. This layer is activated by the predefined linguistic terms—fuzzy sets (A_i, B_j, C_k, etc.) activated by the inputs of the map.

A LINGUISTIC INTERPRETATION OF COMPUTING WITH NEURAL NETWORKS

This style of the usage of fuzzy sets is advantageous in some topologies of neural networks, especially those having a substantial number of outputs and whose learning is carried out in unsupervised form. Fuzzy sets are aimed at the interpretation of results produced by such architectures and facilitates processes of data mining.

To illustrate the idea, we confine ourselves to self-organizing maps. An important property of such architecture is their ability to organize multidimensional patterns in such a way that their vicinity (neighborhood) in the original space is retained when the patterns are mapped onto a certain low-dimensional space so that the map attempts to preserve the main topological properties of the data set. Quite often, the maps are considered in the form of the two-dimensional arrays of regularly distributed processing elements. The mechanism of self-organization is established via competitive learning; the unit that is the "closest" to the actual pattern is given an opportunity to modify its connections and follow the pattern. These modifications are also allowed to affect the neurons situated in the nearest neighborhood of the winning neuron (node) of the map.

Once the training has been completed, the map can locate any multidimensional pattern on the map by identifying the most active processing unit. Subsequently, the linguistic labels are essential components of data mining by embedding the activities of the network in a certain linguistic context.

This concept is visualized in Fig. 3. Let us consider that for each variable we have specified a particular linguistic term (context) defined as a fuzzy set in the corresponding space, namely A_1,A_2, . . . and A_{n1} for x_1, B_1,B_2, . . . and B_{n2} for x_2, etc.

When exposed to any input pattern, the map responds with the activation levels computed at each node in the grid. The logical context leads to an extra two-dimensional grid, the elements of which are activated based on the corresponding activation levels of the nodes located at the lower layer as well as the level of the contexts assumed for the individual variables. These combinations are of the AND form—the upper grid is constructed as a series of the AND neurons. The activation region obtained in this way indicates how much the linguistic description (descriptors)

$$A_i \quad \text{and} \quad B_j \quad \text{and} \quad C_k \quad \text{and} \ldots$$

"covers" (activates) the data space. The higher the activation level of the region the more visible the imposed linguistic pattern within the data set. By performing an analysis of this type for several linguistic data descriptors one can develop a collection of the descriptors that cover the entire data space. We may eventually require that this collection should cover the entire map to a high extent, meaning that

$$\exists_{a>0} \ \forall_{i,j=1,2,n} \ \bigcup_{c \in \mathscr{C}} N(i,j,c) \geq \alpha$$

where $N(i,j,c)$ is the response of the neuron located at the (i,j) and considered (placed) in context c from a certain family of contexts $\mathscr{C}$. Some other criteria could be also anticipated; for example, one may request that the linguistic descriptions are well separated, meaning that their corresponding activation regions in the map are kept almost disjoint.

FUZZY NEURAL COMPUTING STRUCTURES

The examples discussed in the preceding section have revealed a diversity of approaches taken towards building neural network–fuzzy architectures. Taking this into account, we distinguish between two key facets one should take into account in any design endeavor:

- Architectural
- Temporal

These properties are exemplified in the sense of the plasticity and explicit knowledge representation of the resulting neural network–fuzzy structure. The strength of the interaction itself can vary from the level at which the technology of fuzzy sets and neurocomputing are loosely combined and barely coexist to the highest one where there emerges a genuine fusion between the technologies.

Architectures of Fuzzy–Neural Network Systems

The essence of the architectural interaction of fuzzy sets and neural networks is visualized in Fig. 4. This point has already been made clear through the studies in the previous section.

By and large, the role of fuzzy sets gets more visible at the input and output layers of any multilayer structure of the network. The input and output layers are much more oriented toward the capturing the semantics of data rather than focusing on pure numeric processing.

Temporal Aspects of Interaction in Fuzzy–Neural Network Systems

The temporal aspects of interaction arise when dealing with the various levels of intensity of learning, Fig. 4. Again the updates of the connections are much more vigorous at the hidden layers—we conclude that their plasticity (that is an ability to modify the values of the connections) is higher than the others situated close to the input and output layers.

FUZZY NEUROCOMPUTING—AN ARCHITECTURAL FUSION OF FUZZY AND NEURAL NETWORK TECHNOLOGY

In this section we concentrate on a certain category of hybrid processing in which the neurons combine a series of features that are essential to neural networks and symbolic processing. In fact, this is one of the approaches among these reported in the literature (9,10,11). Very often these basic constructs (fuzzy neurons) are exploited as generic building blocks in the development of fuzzy–neural network architectures.

Classes of Fuzzy Neurons

We elaborate on the three models that are representative of most of the existing hybrid architectures as developed in the setting of fuzzy sets and neurocomputing.

The first fuzzy set–oriented construct proposed by Lee and Lee (11) was contrasted with the generic model of the neuron as discussed by McCulloch and Pitts as a binary device. Let us recall that the basic binary neuron has n excitatory inputs $(e_1,e_2,\ldots,e_n)$ and m inhibitory inputs $(i_1,i_2,\ldots,i_m)$. The firing of the neuron is binary: y is set to 1 if all inhibitory inputs are set to 0 and the sum of all excitatory inputs exceeds a threshold level,

$$\sum_{i=1} e_i > T$$

The main generalization proposed by Lee and Lee (11) was to consider that an activity of the neuron is continuous. More specifically, the output of the neuron is one of the positive numbers u_i in [0, 1], $i = 1,2,\ldots,p$, which means that the output (y) reads as

$$y = \begin{cases} u_i & \text{if the neuron is firing} \\ 0 & \text{otherwise} \end{cases}$$

The firing rules are also restated accordingly:

1. All inhibitory inputs are set to 0.
2. The sum of excitatory inputs must be equal or greater than a threshold T.

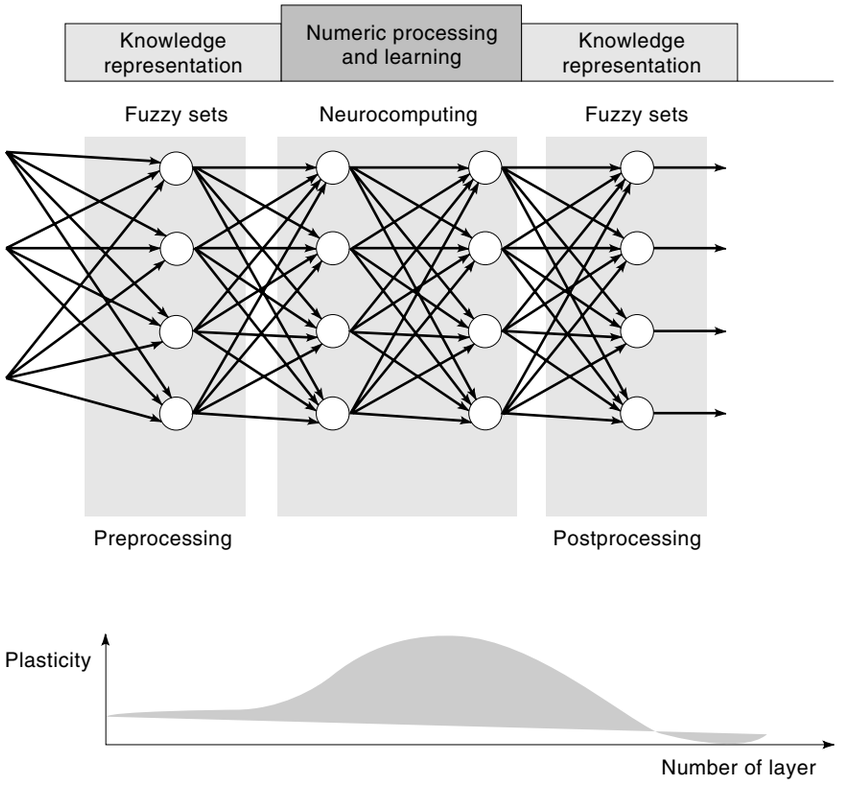

Figure 4. Architectural and temporal synergy of fuzzy set constructs and neural networks. Architectural level: fuzzy sets contribute to the construction of preprocessing and postprocessing modules (input and output layer) and are aimed at knowledge representation whereas numeric processing and learning occurs at the level of the hidden layers of the entire architecture. Temporal interactions: most parametric learning occurs at the level of hidden layers which exhibit high plasticity.

The original study (11) illustrates the application of such fuzzy neurons to a synthesis of fuzzy automata.

An idea proposed by Buckley and Hayashi (12) is to develop a fuzzy neuron in the sense that its connections are viewed as fuzzy sets (more precisely, fuzzy numbers); similarly we consider the inputs to be fuzzy numbers. The underlying formula of the neuron generalizes from the pure numeric neurons and is expressed as

$$Y = f\left(\sum_{i=1}^{n} W_i X_i + \Theta\right)$$

Here the connections and bias and inputs are fuzzy numbers. Moreover, the operations (summation and product) are viewed in terms of fuzzy set operations. Here that the output of the neuron is a fuzzy number. All the operations in the expression that follows are carried out via the extension principle. To illustrate the relevant calculations, let us consider the fuzzy neuron with two inputs (X_1 and X_2). In light of the extension principle, the output of the fuzzy neuron reads as

$$Y(y) = \sup\{\min[X_1(x_1), X_2(x_2), W_1(w_1), W_2(w_2), \Theta(\vartheta)]\}$$

where the supremum in the above expression is taken over all the arguments satisfying the nonlinear constraint

$$y = f\left(\sum_{i=1}^{2} w_i x_i - \mathrm{i} + \vartheta\right)$$

The idea developed by Bortolan (13) makes the previous concept of the fuzzy neuron more computationally attractive by restricting the form of the input fuzzy sets as well as the connections of the neuron to trapezoidal fuzzy sets $\mathcal{T}(x;a,b,c,d)$, see Fig. 5. Such piecewise membership functions represent uncertain variables.

This drastically reduces computational overhead. The pertinent version of a well-known δ learning algorithm is covered in Ref. 13.

Fuzzy Logic Neurons

The main rationale behind this choice (14,15,16) is that the resulting neural networks effortlessly combine learning capabilities with the mechanism of knowledge representation in its *explicit* manner. The neurons are split into two main categories, namely aggregative and referential processing units. We discuss here aggregative neurons. In what follows, we denote t-novus by t (or T). S-novus will be denoted by s (or S).

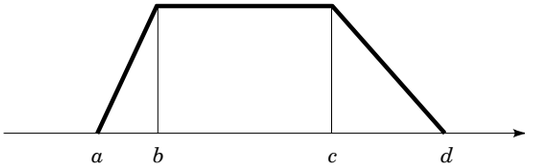

Figure 5. An example of a trapezoidal fuzzy number: a and d denote a lower and upper bound of the linguistic concept. The elements situated in-between b and c belong to the concept at degree 1 and are indistinguishable.

The n-input OR processing unit is governed by the expression

$$y = \mathrm{OR}(\mathbf{x}; \mathbf{w})$$

namely

$$y = \mathop{\mathbf{S}}_{i=1}^{n} (w_i t x_i)$$

The inputs of the neuron are described by $\mathbf{x}$ while the vector $\mathbf{w}$ summarizes its connections. The computations of the output (y) rely on the use of some triangular norms (s and t norm). Observe that if $\mathbf{w} = \mathbf{1}$ then $y = \mathrm{OR}(x_1,x_2,\ldots,x_n)$ so that the neuron reduces to a standard OR gate encountered in digital logic. The AND neuron is described in the following form

$$y = \mathrm{AND}(\mathbf{x}; \mathbf{w})$$

or equivalently

$$y = \mathop{\mathbf{T}}_{i=1}^{n} (w_i s x_i)$$

Note that the composition operation used here uses the s and t norm in a reversed order.

An important class of fuzzy neural networks concerns an approximation of mappings between the unit hypercubes (namely, from $[0,1]^n$ to $[0,1]^m$ or $[0,1]$ for $m = 1$). These mappings are realized in a logic-based format. To fully comprehend the fundamental idea behind this architecture, let us note some very simple yet powerful concepts form the realm of two-valued systems. The well-known Shannon's theorem states that any Boolean function $\{0,1\}^n \rightarrow \{0,1\}$ can be uniquely represented as a logical sum (union) of minterms (a so-called SOM representation) or, equivalently, a product of some maxterms (known as a POM representation). By *minterm* we mean an AND combination of all the input variables of this function; they could appear either in a direct or complemented (negated) form. Similarly, the *maxterm* consists of the variables that now occur in their OR combination. A complete list of minterms and maxterms for Boolean functions of two variables consists of the expressions

$$\overline{x}_1 \text{ AND } \overline{x}_2, x_1 \text{ AND } \overline{x}_2, \overline{x}_1 \text{ AND } x_2, x_1 \text{ AND } x_2 \text{ for minterms}$$

$$\overline{x}_1 \text{ OR } \overline{x}_2, x_1 \text{ OR } \overline{x}_2, \overline{x}_1 \text{ OR } x_2, x_1 \text{ OR } x_2 \text{ for maxterms}$$

From a functional point of view, the minterms can be identified with the AND neurons while the OR neurons can be used to produce the corresponding maxterms. It is also noticeable that the connections of these neurons are restricted to the two-valued set $\{0,1\}$, therefore making these neurons two-valued selectors. Taking into account the fundamental representation of the Boolean functions, two complementary (dual) architectures are envisioned. In the first case, the network includes a single hidden layer that is constructed with the aid of the AND neurons and the output layer consisting of the OR neurons (SOM version of the network). The dual type of the network is of the POM type in which the hidden layer consists of some OR neurons while the output layer is formed by the AND neurons.

Two points are worth making here that contrast between the logic processors (LP) in their continuous and two-valued versions:

1. The logic processor *represents* or *approximates* data. For the Boolean data, assuming that all the input combinations are different, we are talking about a *representation* of the corresponding Boolean function. In this case the POM and SOM versions of the logic processors for the same Boolean function are equivalent.
2. The logic processor used for the continuous data approximates a certain unknown fuzzy function. The equivalence of the POM and SOM types of the obtained logic processors is not guaranteed at all. Moreover, the approximation exhibits an inherent logical flavor not necessarily leading to the same approximation accuracy as achieved for "classic" neural networks. This should not be regarded as a shortcoming, as in return we obtain some essential transparency of the neural architecture that could be easily interpreted in the form of "if–then" statements—the most evident enhancement of the architecture in an attempt to alleviate the black box nature inherent of most of the neural networks.

CONCLUSIONS

Fuzzy sets and neurocomputing are two supplementary technologies. The two-way integration is not only possible but highly beneficial. The knowledge-based faculties are well handled by the technology of fuzzy sets, while the learning activities are chiefly addressed by neural networks. Interestingly, there are a number of new constructs combining the ideas stemming from fuzzy sets and neural networks. We have investigated various levels of synergy and proposed a consistent classification of the systems emerging as an outcome of the symbiosis of these two technologies.

BIBLIOGRAPHY

1. D. E. Rumelhart and J. L. McLelland, *Parallel Distributed Processing,* Cambridge, MA: MIT Press, 1986.
2. L. A. Zadeh, Fuzzy sets and information granularity. In: M. M. Gupta, R. K. Ragade, and R. R. Yager (eds.), *Advances in Fuzzy Set Theory and Applications,* Amsterdam: North Holland, 1979, pp. 3–18.
3. W. Pedrycz, Selected issues of frame of knowledge representation realized by means of linguistic labels, *Int. J. Intelligent Systems,* **7**:155–170, 1992.
4. S. Chen, C. F. N. Cowan, and P. M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Trans. Neural Networks,* **2**:302–309 1991.
5. J. Moody and C. Darken, Fast learning networks of locally-tuned processing units, *Neural Computing,* **1**:281–294, 1989.
6. D. Dubois and H. Prade, *Possibility Theory—An Approach to Computerized Processing of Uncertainty,* New York: Plenum Press, 1988.
7. F. M. Silva and L. B. Almeida, Acceleration techniques for the back-propagation algorithm. In *Lecture Notes in Computer Science,* Berlin: Springer-Verlag, 1990, Vol. 412, pp. 110–119.
8. J. M. Keller and D. J. Hunt, Incorporating fuzzy membership functions into the perceptron algorithm, *IEEE Trans. Pattern Anal. Mach. Intell.,* **PAMI-7**:693–699, 1985.
9. H. Ishibuchi, R. Fujioka, and H. Tanaka, An architecture of neural networks for input vectors of fuzzy numbers. *Proc. IEEE Int. Conf. Fuzzy Systems* (FUZZ-IEEE '92), San Diego, March 8–12, 1992, pp. 1293–1300.
10. I. Requena and M. Delgado, R-FN: A model of fuzzy neuron, *Proc. 2nd Int. Conf. Fuzzy Logic Neural Networks* (IIZUKA '92), Iizuka, Japan, July 17–22, 1992, pp. 793–796.
11. S. C. Lee and E. T. Lee, Fuzzy neural networks, *Math Biosci.* **23**:151–177, 1975.
12. J. Buckley and Y. Hayashi, Fuzzy neural networks: A survey, *Fuzzy Sets Systems* **66**:1–14, 1994.
13. G. Bortolan, Neural networks for the processing of fuzzy sets. In: M. Marinaro and P. G. Morasso (eds.), *Proceeding of the International Conference on Artificial Neural Networks,* London: Springer-Verlag, 1994, pp. 181–184.
14. W. Pedrycz, Fuzzy neural networks and neurocomputations. *Fuzzy Sets Systems* **56**:1–28, 1993.
15. W. Pedrycz, *Fuzzy Sets Engineering.* Boca Raton, FL: CRC, 1995.
16. W. Pedrycz and A. F. Rocha, Fuzzy-set based models of neurons and knowledge-based networks, *IEEE Trans. Fuzzy Systems,* **1**:254–266, 1993.

WITOLD PEDRYCZ
University of Manitoba